공학석사학위논문

FALCON: FAst and Lightweight CONvolution for Compressing and Accelerating Convolutional Neural Networks

2019년 7월

서울대학교 대학원
컴퓨터공학부
Chun Quan

Abstract

FALCON: FAst and Lightweight CONvolution for Compressing and Accelerating Convolutional Neural Networks

Chun Quan
Department of Computer Science & Engineering
The Graduate School
Seoul National University

How can we efficiently compress Convolution Neural Networks (CNN) while maintaining the accuracy of classification tasks? One of the promising approaches is based on depthwise separable convolution which replaces a standard convolution with a depthwise convolution and pointwise convolution. However, previous works based on the depthwise separable convolution are limited since 1) they are mostly heuristic approaches without precise understanding of their relations to the standard convolution, and 2) their accuracies cannot match that of the standard convolution.

In this paper, we propose FALCON, an accurate and lightweight method for compressing CNN. FALCON is derived by interpreting existing convolution methods based on depthwise separable convolution using EHP, our proposed mathematical formulation to approximate the standard convolution ker-

nel. Such interpretation leads to developing a generalized version rank-k FALCON which further improves the accuracy while sacrificing a bit of compression and computation. Experiments show that FALCON outperforms 1) existing methods based on depthwise separable convolution, and 2) the standard CNN model by up to 8× compression and 8× computation reduction while ensuring similar accuracy. We also demonstrate that rank-k FALCON provides even better accuracy than the standard convolution in many cases, while using smaller numbers of parameters and floating point operations.

Keywords : CNN compression, CNN acceleration, convolution

Student Number : 2017-26296

Contents

I. Introduction **1**

II. Preliminaries **4**

2.1 Convolution Neural Network 4

2.2 Depthwise Separable Convolution 6

2.3 Methods Based on Depthwise Separable Convolution 9

2.4 Hadamard Product 10

III. Proposed Method **12**

3.1 Extended Hadamard Product (EHP) 12

3.2 Depthwise Separable Convolution and EHP 14

3.3 FAst and Lightweight CONvolution (FALCON) 16

3.4 Rank-k FALCON 19

3.5 Quantitative Analysis 20

3.5.1 FALCON 20

3.5.2 Rank-k FALCON 22

IV. Experiments **23**

4.1 Experimental Setup 23

4.2 Fitting Convolution Unit into Model 25

4.3 Accuracy vs. Compression 32

4.4 Accuracy vs. Computation 32

4.5 Rank-k FALCON 33

V. Related Works **36**

VI. Conclusion **38**

References **39**

Appendix **43**

A Generality of EHP 43

B Parameters and FLOPs 44

Acknowledgements **45**

List of Figures

Figure 1. Convolution operation in convolution layer. 5

Figure 2. Comparison of architectures. BN denotes batch-normalization. Relu and Relu6 are activation functions. (a) Standard convolution. (b) Our proposed method FALCON. (c) Depthwise separable convolution (DSConv) in Mobilenet. (d) MobileConvV2 used in MobilenetV2. (e) ShuffleUnit used in Shufflenet. 8

Figure 3. Relation between standard convolution and depthwise separable convolution expressed with EHP. The common axes correspond to the input channel-axis of standard convolution. 14

Figure 4. Relation between standard convolution and FALCON expressed with EHP. The common axes correspond to the output channel-axis of standard convolution. $TT_{(1,2,4,3)}$ indicates tensor transpose operation to permute the third and the fourth dimensions of a tensor. 16

Figure 5. Relation between the standard convolution and rank-k FALCON expressed with EHP. $TT_{(1,2,4,3)}$ indicates tensor transpose operation to permute the third and the fourth dimensions of a tensor. 19

Figure 6. Accuracy w.r.t. number of parameters on different models and datasets. The three blue circles correspond to rank-1, 2, 3 FALCON (from left to right order), respectively. FALCON provides the best accuracy for a given number of parameters. 30

Figure 7. Accuracy w.r.t. FLOPs on different models and datasets. The three blue circles correspond to rank-1, 2, 3 FALCON (from left to right order), respectively. FALCON provides the best accuracy for a given number of FLOPs. 31

List of Tables

Table 1. Symbols. 7
Table 2. Datasets. 24
Table 3. Performance of FALCON compared to StConv, DSConv, MobileConvV2, and ShuffleUnit on CIFAR10/CIFAR100 datasets. Bold font indicates the best accuracy among competing compression methods. FALCON gives the highest accuracy with the similar number of parameters and FLOPs compared to other methods. 26
Table 4. Performance of FALCON compared to StConv, DSConv, MobileConvV2, and ShuffleUnit on SVHN dataset. Bold font indicates the best accuracy among competing compression methods. FALCON gives the highest accuracy with the similar number of parameters and FLOPs compared to other methods. 27
Table 5. Performance of FALCON compared to StConv, DSConv, MobileConvV2, and ShuffleUnit on ImageNet dataset. Bold font indicates the best accuracy among competing compression methods. FALCON provides the highest accuracy with similar number of parameters and FLOPs compared to other methods. 28
Table 6. Performance of rank-k FALCON on CIFAR10/CIFAR100 datasets. 34
Table 7. Performance of rank-k FALCON on SVHN dataset. 35

Table 8. Performance of rank-k FALCON on ImageNet dataset. 35
Table 9. the numbers of parameters and FLOPs of FALCON and competitors. Symbols are described in Table 1. 44

Chapter 1

Introduction

How can we efficiently reduce the size and the energy consumption of Convolutional Neural Networks (CNN) while maintaining the accuracy of classification tasks? Nowadays, CNN is widely used in various areas including computer vision [1, 2, 3], natural language processing [4], recommendation system [5], etc. In addition, model compression becomes an important technique as the model capacity and the number of parameters in CNN have increased continuously. One of the recent and promising approaches for compressing CNN is depthwise separable convolution [6] which replaces a standard convolution with depthwise and pointwise convolutions. The depthwise convolution applies separate 2D convolution kernel for each input channel, and the pointwise convolution changes the channel size using 1×1 convolution (details in Section 2.2). Several recent methods [7, 8, 9] based on the depthwise separable convolution show reasonable performances in terms of compression and computation reduction.

However, existing approaches based on the depthwise separable convolution have several crucial limitations. First, they are heuristic methods, and the relation between their method and the standard convolution is not clearly identified. Second, due to the heuristic nature of the methods, generalizing the method is difficult. Third, although they give reasonable compression and computation reduction, their accuracies are not sufficient compared to the standard convolution based model.

In this paper, we propose FALCON, an accurate and lightweight method for compressing CNN. FALCON overcomes the limitation of the previous methods based on the depthwise separable convolution using the following two main ideas. First, we precisely define the relation between the standard convolution and the depthwise separable convolution using EHP (Extended Hadamard Product), our proposed mathematical formulation to correlate the standard convolution kernel with the depthwise convolution kernel and the pointwise convolution kernel. We then design FALCON by fine-tuning and reordering the results of EHP to improve the accuracy of convolution operation. Second, based on the precise definition, we generalize the FALCON to design rank-k FALCON, which further improves accuracy while sacrificing a bit of computation and compression reductions. As a result, FALCON provides a superior accuracy compared to other methods based on depthwise separable convolution, with similar compression and computation rates, and rank-k FALCON further improves accuracy outperforming even the original convolution in many cases. Our contributions are summarized as follows:

- **Generalization.** We analyze and generalize depthwise separable convolution to our proposed EHP (Extended Hadamard Product) operation. Such generalization enables to precisely understand the relation between depthwise separable convolution and standard convolution. Furthermore, the generalization leads to our proposed method FALCON by fine-tuning operations.
- **Algorithm.** We propose FALCON, a CNN compression method based on the depthwise separable convolution. FALCON is carefully designed to compress CNN with little loss of accuracy. We further propose rank-k

FALCON to further improve accuracy with a little sacrifice in compression and computation rates. We also theoretically analyze the compression and computation reduction of FALCON and other competitors.

- **Experiments.** We perform extensive experiments and show that FALCON 1) outperforms other state-of-the-art methods based on depthwise separable convolution for compressing CNN, and 2) provides up to 8× compression and computation reduction compared to the standard convolution while giving similar accuracies. Furthermore, we show that rank-k FALCON provides even better accuracy than the standard convolution in many cases while using smaller number of parameters and floating point operations.

The rest of this paper is organized as follows. Chapter 2 explains preliminaries. Chapter 3 describes our proposed method FALCON. Chapter 4 presents experimental results. After discussing related works in Chapter 5, we conclude in Chapter 6.

Chapter 2

Preliminaries

We describe preliminaries on CNN, depthwise separable convolution, methods based on depthwise separable convolution, and Hadamard product. Table 1 lists the symbols used in this paper.

2.1 Convolution Neural Network

Convolution Neural Network (CNN) is a type of deep neural network used mainly for structured data. CNN uses convolution operation in convolution layers. In the following, we discuss CNN when applied to typical image data with RGB channels.

Each convolution layer has three components: input feature maps, convolution kernel, and output feature maps. The input feature maps $\mathcal{I} \in \mathbb{R}^{H \times W \times M}$ and the output feature maps $\mathcal{O} \in \mathbb{R}^{H' \times W' \times N}$ are 3-dimensional tensors, and the convolution kernel $\mathcal{K} \in \mathbb{R}^{D \times D \times M \times N}$ is a 4-dimensional tensor.

The convolution operation is defined as:

$$\mathcal{O}_{h',w',n} = \sum_{i=1}^{D} \sum_{j=1}^{D} \sum_{m=1}^{M} \mathcal{K}_{i,j,m,n} \cdot \mathcal{I}_{h_i,w_j,m} \tag{2.1}$$

where the relations between height h_i and width w_j of input, and height h' and width w' of output are as follows:

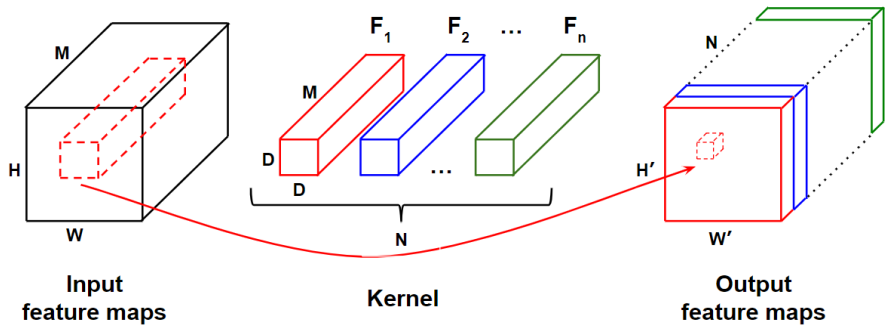


Figure 1: Convolution operation in convolution layer.

$$h_i = (h' - 1)s + i - p \quad \text{and} \quad w_j = (w' - 1)s + j - p \tag{2.2}$$

where s is the stride size, and p is the padding size. The third and the fourth dimensions of the convolution kernel $\mathcal{K}$ must match the number M of input channels, and the number N of output channels, respectively.

Convolution kernel $\mathcal{K}$ can be seen as N 3-dimensional filters $\mathcal{F}_n \in \mathbb{R}^{D \times D \times M}$. As shown in Figure 1, each filter $\mathcal{F}_n$ in kernel $\mathcal{K}$ performs convolution operation while sliding over all spatial locations on input feature maps. Each filter produces one output feature map.

Standard convolution needs D^2MN parameters and $H'W'D^2MN$ floating point operations (FLOPs). In this paper, we define FLOPs as the number of multiply-adds.

2.2 Depthwise Separable Convolution

Depthwise Separable Convolution (DSConv) consists of two sub-layers: depthwise convolution and pointwise convolution. Depthwise convolution (DWConv) kernel consists of several $D \times D$ 2-dimensional filters. The number of 2-dimension filters is the same as the number of input feature maps. Each filter is applied on the corresponding input feature map, and produces an output feature map. Pointwise convolution (PWConv), also known as 1×1 convolution, is a standard convolution with kernel size 1.

DSConv is defined as follows:

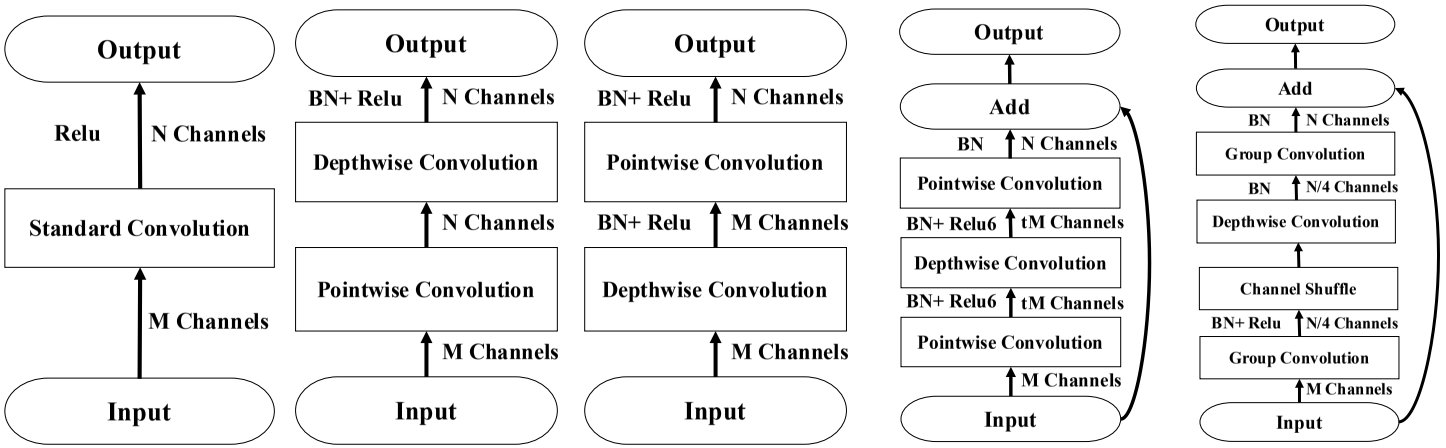
$$\mathfrak{O}'_{h',w',m} = \sum_{i=1}^{D}\sum_{j=1}^{D} \mathfrak{D}_{i,j,m} \cdot \mathfrak{I}_{h_i,w_j,m} \tag{2.3}$$

$$\mathfrak{O}_{h',w',n} = \sum_{m=1}^{M} \mathbf{P}_{m,n} \cdot \mathfrak{O}'_{h',w',m} \tag{2.4}$$

where $\mathfrak{D}_{i,j,m}$ and $\mathbf{P}_{m,n}$ are depthwise convolution kernel and pointwise convolution kernel, respectively. $\mathfrak{O}'_{h',w',m} \in \mathbb{R}^{H' \times W' \times M}$ denotes intermediate feature maps. DSConv performs DWConv on input feature maps $\mathfrak{I}_{h_i,w_j,m}$ using Equation (2.3), and generates intermediate feature maps $\mathfrak{O}'_{h',w',m}$. Then, DSConv performs PWConv on $\mathfrak{O}'_{h',w',m}$ using Equation (2.4), and generates output feature maps $\mathfrak{O}_{h',w',n}$.

Table 1: Symbols.

Symbol	Description
$\mathcal{K}$	convolution kernel of size $\mathbb{R}^{D\times D\times M\times N}$
$\mathcal{I}$	input feature maps of size $\mathbb{R}^{H\times W\times M}$
$\mathcal{O}$	output feature maps of size $\mathbb{R}^{H'\times W'\times N}$
D	height and width of kernel (kernel size)
M	number of input feature map (input channels)
N	number of output feature map (output channels)
H	height of input feature map
W	width of input feature map
H'	height of output feature map
W'	width of output feature map
s	stride
p	padding
$\odot_{p,q}$	Extended Hadamard Product (EHP)
t	expansion ratio in MobilenetV2
g	number of groups in Shufflenet



(a) Standard Convolution (b) FALCON (proposed) (c) Mobilenet [7] (d) MobilenetV2 [8] (e) Shufflenet [9]

Figure 2: Comparison of architectures. BN denotes batch-normalization. Relu and Relu6 are activation functions. (a) Standard convolution. (b) Our proposed method FALCON. (c) Depthwise separable convolution (DSConv) in Mobilenet. (d) MobileConvV2 used in MobilenetV2. (e) ShuffleUnit used in Shufflenet.

Howard et al. [7] used DSConv as the basic convolution block in their Mobilenet model. The architecture of each convolution layer in DSConv is illustrated in Figure 2(c). DSConv can efficiently replace $D \times D$ standard convolution with only $D^2M + MN$ parameters and $HWD^2M + H'W'MN$ FLOPs.

2.3 Methods Based on Depthwise Separable Convolution

We discuss recent CNN methods based on Depthwise Separable Convolution. DSConv was first introduced by Sifre [6]. Chollet et al. [10] built Xception module using DWConv in a few layers. Howard et al. [7] built Mobilenet with all convolution layers replaced by DWConv.

MobilenetV2 Sandler et al. [8] proposed a new convolution architecture which we call as MobileConvV2, in their MobilenetV2 model. MobileConvV2 consists of three sub-layers as shown in Figure 2(d). The first and the third sub-layers are pointwise convolution for adjusting the number of channels. The first sub-layer expands the number of channels from M to tM, where t is an expansion ratio. The second sub-layer is a $D \times D$ depthwise convolution. Since depthwise convolution cannot change the number of channels, the third sub-layer adjusts the number of channels from tM to N. There is a shortcut connection between the input, and the output of the third sub-layer to facilitate flow of gradient across multiple layers. MobileConvV2 needs $tM^2 + D^2tM + tMN$ parameters and $tHWM^2 + tH'W'D^2M + tH'W'MN$ FLOPs.

Shufflenet Zhang et al. [9] proposed a computation-efficient CNN architecture named Shufflenet. As shown in Figure 2(e) each unit of Shufflenet (we call it ShuffleUnit) consists of three sublayers, first group pointwise convolution, depthwise convolution, and second group pointwise convolution, as well as a shortcut. The number of depthwise convolution channels is $\frac{1}{4}$ of output channels N. ShuffleUnit uses group convolution in two pointwise convolution layers to reduce the parameters and FLOPs. However, it is hard to exchange information among groups when group convolutions are stacked. To deal with this problem, ShuffleUnit adds a channel shuffle layer after the first pointwise group convolution. The channel shuffle layer rearranges the order of channels by transposing the output channel dimensions, making it possible to obtain information from different groups. The number of groups is represented as g. ShuffleUnit needs $\frac{1}{4g}MN+\frac{1}{4}D^2N+\frac{1}{4g}N^2$ parameters and $\frac{1}{4g}HWMN+\frac{1}{4}H'W'D^2N+\frac{1}{4g}H'W'N^2$ FLOPs.

2.4 Hadamard Product

Hadamard product is defined as the element-wise product of two matrices with the same shape. Given two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{I \times J}$ with the same dimension, Hadamard product ($\odot$) is defined as follows:

$$(\mathbf{A} \odot \mathbf{B})_{i,j} = \mathbf{A}_{i,j} \cdot \mathbf{B}_{i,j}$$

The product can be generalized from matrices to tensors. Given two ten-

sors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^{I \times J \times \ldots \times N}$, Hadamard product ($\odot$) is defined as follows:

$$(\boldsymbol{\mathcal{A}} \odot \boldsymbol{\mathcal{B}})_{i,j,\ldots,n} = \boldsymbol{\mathcal{A}}_{i,j,\ldots,n} \cdot \boldsymbol{\mathcal{B}}_{i,j,\ldots,n}$$

Chapter 3

Proposed Method

We describe FALCON, our proposed method for compressing CNN. We first define Extended Hadamard Product (EHP), a key mathematical formulation to generalize depthwise separable convolution in Section 3.1. We interpret depthwise separable convolution used in Mobilenet using EHP in Section 3.2. We propose FALCON in Section 3.3 and explain why FALCON can replace standard convolution. Then, we propose rank-k FALCON which extends the basic FALCON in Section 3.4. Finally, we theoretically analyze the performance of FALCON in Section 3.5.

3.1 Extended Hadamard Product (EHP)

We define Extended Hadamard Product (EHP), a generalized elementwise product for two operands of different shapes to generalize the formulation of the relation between standard convolution and depthwise separable convolution.

Before generalizing the formulation, we give an example of formulating the relation between standard convolution and depthwise separable convolution. Suppose we have a 4-order standard convolution kernel $\mathcal{K} \in \mathbb{R}^{I \times J \times M \times N}$, a 3-order depthwise convolution kernel $\mathcal{D} \in \mathbb{R}^{I \times J \times M}$, and a pointwise convolution kernel $\mathbf{P} \in \mathbb{R}^{M \times N}$. Let $\mathcal{K}_{i,j,m,n}$ be (i,j,m,n)-th element of $\mathcal{K}$, $\mathcal{D}_{i,j,m}$ be (i,j,m)-th element of $\mathcal{D}$, and $\mathbf{P}_{m,n}$ be (m,n)-th element of $\mathbf{P}$. Then, the

relation between $\mathcal{K}_{i,j,m,n}$, $\mathcal{D}_{i,j,m}$, and $\mathbf{P}_{m,n}$ is

$$\mathcal{K}_{i,j,m,n} = \mathcal{D}_{i,j,m} \cdot \mathbf{P}_{m,n}$$

To formally express this relation, we define Extended Hadamard Product (EHP) as follows.

Definition 1 (Extended Hadamard Product) Given p-order tensor $\mathcal{D} \in \mathbb{R}^{I_1 \times \cdots \times I_{p-1} \times M}$ and q-order tensor $\mathcal{P} \in \mathbb{R}^{M \times J_1 \times \cdots \times J_{q-1}}$, the Extended Hadamard Product $\mathcal{D} \odot_E \mathcal{P}$ of $\mathcal{D}$ and $\mathcal{P}$ is defined to be the tensor $\mathcal{K} \in \mathbb{R}^{I_1 \times \ldots \times I_{p-1} \times M \times J_1 \times \ldots \times J_{q-1}}$ where the last axis of $\mathcal{D}$ and the first axis of $\mathcal{P}$ are the common axes such that

$$\mathcal{K}_{i_1,\ldots,i_{p-1},m,j_1,\ldots,j_{q-1}} = \mathcal{D}_{i_1,\ldots,i_{p-1},m} \cdot \mathcal{P}_{m,j_1,\ldots,j_{q-1}}$$

for all elements of $\mathcal{K}$. □

Contrary to Hadamard Product which is defined only if the shapes of the two operands are the same, Extended Hadamard Product (EHP) deals with tensors of different shapes. Now, we define a special case of Extended Hadamard Product (EHP) for a third-order tensor and a matrix.

Definition 2 (EHP for a third order tensor and a matrix) Given a third-order tensor $\mathcal{D} \in \mathbb{R}^{I \times J \times M}$ and a matrix $\mathbf{P} \in \mathbb{R}^{M \times N}$, the Extended Hadamard Product $\mathcal{D} \odot_E \mathbf{P}$ is defined to be the tensor $\mathcal{K} \in \mathbb{R}^{I \times J \times M \times N}$ where the third axis of the tensor $\mathcal{D}$ and the first axis of the matrix $\mathbf{P}$ are the common axes such that

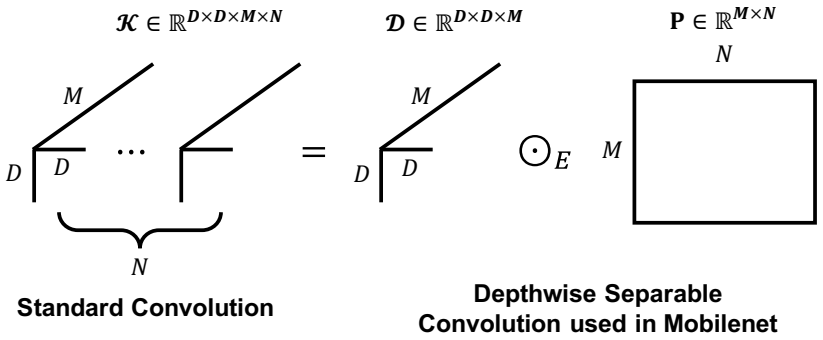


Figure 3: Relation between standard convolution and depthwise separable convolution expressed with EHP. The common axes correspond to the input channel-axis of standard convolution.

$$\mathcal{K}_{i,j,m,n} = \mathcal{D}_{i,j,m} \cdot \mathbf{P}_{m,n}.$$

for all elements of $\mathcal{K}$. □

We will see that the depthwise separable convolution in Mobilenet is easily expressed with EHP in Section 3.2; we also propose a new architecture Falcon based on EHP in Section 3.3. EHP is also a core operation that helps understanding other convolution architectures including MobilenetV2 and Shufflenet (see Appendix A).

3.2 Depthwise Separable Convolution and EHP

In this section, we discuss how to represent the convolution layer of Mobilenet as Extended Hadamard Product (EHP) described in Section 3.1. We

interpret the depthwise separable convolution which is the convolution of Mobilenet as an application of EHP. This interpretation leads to designing a better convolution architecture FALCON in Section 3.3.

We represent the relation between standard convolution kernel $\mathcal{K} \in \mathbb{R}^{D\times D\times M\times N}$ and depthwise separable convolution consisting of depthwise convolution kernel $\mathcal{D} \in \mathbb{R}^{D\times D\times M}$ and pointwise convolution kernel $\mathbf{P} \in \mathbb{R}^{M\times N}$ using one EHP operation. Figure 3 illustrates the relation between standard convolution and depthwise separable convolution used in Mobilenet.

We show that applying depthwise separable convolution with $\mathcal{D}$ and $\mathbf{P}$ is equivalent to applying standard convolution with a kernel $\mathcal{K}$ which is constructed from $\mathcal{D}$ and $\mathbf{P}$.

Theorem 1 *Applying depthwise separable convolution with depthwise convolution kernel $\mathcal{D} \in \mathbb{R}^{D\times D\times M}$ and pointwise convolution kernel $\mathbf{P} \in \mathbb{R}^{M\times N}$ is equivalent to applying standard convolution with kernel $\mathcal{K} = \mathcal{D} \odot_E \mathbf{P}$.*

Proof 3.2.1 *From the definition of EHP, $\mathcal{K}_{i,j,m,n} = \mathcal{D}_{i,j,m} \cdot \mathbf{P}_{m,n}$. Based on Equation (2.1), we replace the kernel $\mathcal{K}_{i,j,m,n}$ with the depthwise convolution kernel $\mathcal{D}_{i,j,m}$ and the pointwise convolution kernel $\mathbf{P}_{m,n}$.*

$$\mathcal{O}_{h',w',n} = \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{m=1}^{M} \mathcal{D}_{i,j,m} \cdot \mathbf{P}_{m,n} \cdot \mathcal{I}_{h_i,w_j,m}$$

where $\mathcal{I}_{h_i,w_j,m}$ is the (h_i, w_j, m)-th entry of the input. We split the above equa-

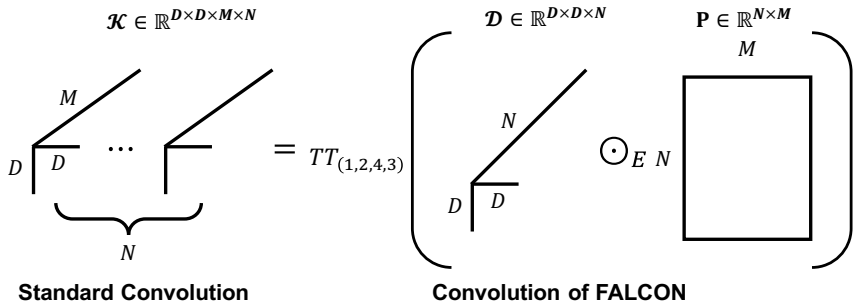


Figure 4: Relation between standard convolution and FALCON expressed with EHP. The common axes correspond to the output channel-axis of standard convolution. $TT_{(1,2,4,3)}$ indicates tensor transpose operation to permute the third and the fourth dimensions of a tensor.

tion into the following two equations.

$$\mathfrak{O}'_{h',w',m} = \sum_{i=1}^{D}\sum_{j=1}^{D} \mathfrak{D}_{i,j,m} \cdot \mathfrak{I}_{h_i,w_j,m} \tag{3.1}$$

$$\mathfrak{O}_{h',w',n} = \sum_{m=1}^{M} \mathbf{P}_{m,n} \cdot \mathfrak{O}'_{h',w',m} \tag{3.2}$$

where $\mathfrak{O}'_{h',w',m} \in \mathbb{R}^{H'\times W'\times M}$ *is an intermediate tensor. Note that Equations* (3.1) *and* (3.2) *correspond to the depthwise convolution and the pointwise convolution, respectively. Therefore, the output* $\mathfrak{O}'_{h',w',m}$ *is equal to the output applying depthwise separable convolution used in Mobilenet.* □

3.3 FAst and Lightweight CONvolution (Falcon)

We propose FALCON (FAst and Lightweight CONvolution), a novel lightweight convolution that replaces standard convolution. FALCON is an efficient method with fewer parameters and computations than the standard convolution re-

quires. In addition, FALCON has better accuracy than competitors while having similar compression and computation rates.

We observe that a typical convolution has more output channels than input channels. In such setting, performing depthwise convolution after pointwise convolution would enrich feature space to extract more features from the depthwise convolution; on the other hand, performing pointwise convolution after depthwise convolution as in Mobilenet only combines features extracted from a limited feature space.

Based on the observation, FALCON first applies pointwise convolution to generate an intermediate tensor $\mathcal{O}' \in \mathbb{R}^{H \times W \times N}$, and then applies depthwise convolution.

We represent the relation between standard convolution kernel $\mathcal{K} \in \mathbb{R}^{D \times D \times M \times N}$ and FALCON using EHP operation of pointwise convolution kernel $\mathbf{P} \in \mathbb{R}^{N \times M}$ and depthwise convolution kernel $\mathcal{D} \in \mathbb{R}^{D \times D \times N}$ in Figure 4. In FALCON, the kernel $\mathcal{K}$ is represented by EHP of $\mathcal{D}$ and $\mathbf{P}$ as follows:

$$\mathcal{K} = TT_{(1,2,4,3)}(\mathcal{D} \odot_E \mathbf{P}) \quad \text{s.t.} \quad \mathcal{K}_{i,j,m,n} = \mathbf{P}_{n,m} \cdot \mathcal{D}_{i,j,n}.$$

where $TT_{(1,2,4,3)}$ indicates tensor transpose operation to permute the third and the fourth dimensions of a tensor. Note that the common axis is the output channel axis of the standard convolution, unlike EHP for depthwise separable convolution where the common axis is the input channel axis of the standard convolution.

As in Section 3.2, we describe that applying FALCON is equivalent to applying standard convolution.

Theorem 2 FALCON *which applies pointwise convolution with kernel* $\mathbf{P} \in \mathbb{R}^{N \times M}$ *and then depthwise convolution with kernel* $\mathfrak{D} \in \mathbb{R}^{D \times D \times N}$ *is equivalent to applying standard convolution with kernel* $\mathfrak{K} = TT_{(1,2,4,3)}(\mathfrak{D} \odot_E \mathbf{P})$.

Proof 3.3.1 *From Equation* (2.1)*, we replace the kernel* $\mathfrak{K}_{i,j,m,n}$ *with the pointwise convolution kernel* $\mathbf{P}$ *and the depthwise convolution kernel* $\mathfrak{D}$.

$$\mathfrak{O}_{h',w',n} = \sum_{m=1}^{M} \sum_{i=1}^{D} \sum_{j=1}^{D} \mathbf{P}_{m,n} \cdot \mathfrak{D}_{i,j,n} \cdot \mathfrak{I}_{h_i,w_j,m}$$

where $\mathfrak{I}_{h_i,w_j,m}$ *is the* (h_i, w_j, m)*-th entry of the input* $\mathfrak{I}$*. We split the above equation into the following two equations.*

$$\mathfrak{O}'_{h_i,w_j,n} = \sum_{m=1}^{M} \mathbf{P}_{m,n} \cdot \mathfrak{I}_{h_i,w_j,m} \tag{3.3}$$

$$\mathfrak{O}_{h',w',n} = \sum_{i=1}^{D} \sum_{j=1}^{D} \mathfrak{D}_{i,j,n} \cdot \mathfrak{O}'_{h_i,w_j,n} \tag{3.4}$$

where $\mathfrak{I}$*,* $\mathfrak{O}'$*, and* $\mathfrak{O}$ *are the input, the intermediate, and the output tensors of convolution layer, respectively. Note that Equations* (3.3) *and* (3.4) *correspond to pointwise convolution and depthwise convolution, respectively. Therefore, the output* $\mathfrak{O}'_{h',w',n}$ *is equal to the output applying* FALCON. □

After pointwise convolution and depthwise convolution, we add batch-normalization and ReLU activation function as shown in Figure 2(b). We note that FALCON significantly reduces the numbers of parameters and FLOPs compared to standard convolution, which we discuss at Section 3.5.

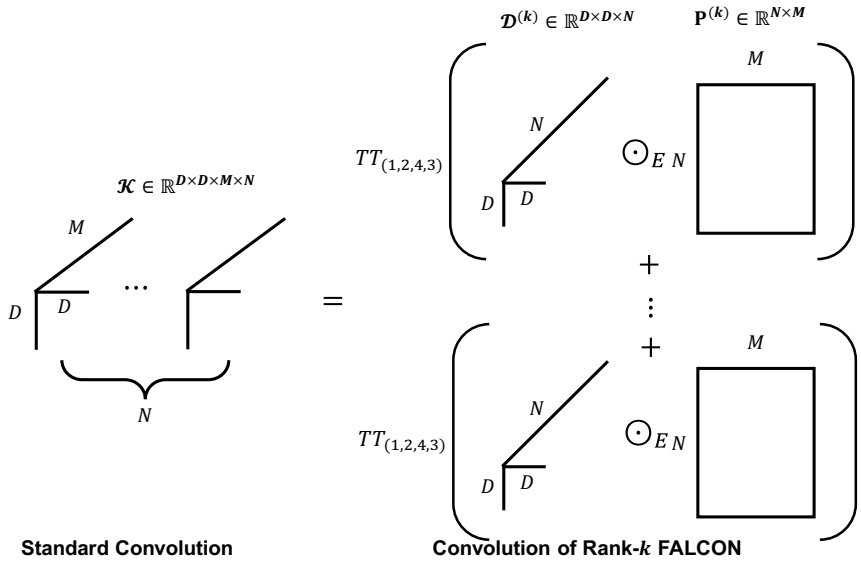


Figure 5: Relation between the standard convolution and rank-k FALCON expressed with EHP. $TT_{(1,2,4,3)}$ indicates tensor transpose operation to permute the third and the fourth dimensions of a tensor.

3.4 Rank-k Falcon

We propose rank-k FALCON, an extended version of FALCON to improve accuracy while sacrificing a bit of compression and computation reduction. The main idea is to perform k independent FALCON operations, and sum up the result. After that, we apply batch-normalization (BN) and ReLU activation function to the summed result. Since each FALCON operation requires independent parameters for pointwise convolution and depthwise convolution, the number of parameters increases and thus the compression and the computation rates decrease; however, it improves accuracy by enlarging the model capacity. We formally define the rank-k FALCON with EHP as follows.

Definition 3 (Rank-k Falcon with Extended Hadamard Product) *Rank-*

k FALCON *expresses standard convolution kernel* $\mathcal{K} \in \mathbb{R}^{D\times D\times M\times N}$ *as EHP of depthwise convolution kernel* $\mathcal{D}^{(k)} \in \mathbb{R}^{D\times D\times N}$ *and pointwise convolution kernel* $\mathbf{P}^{(k)} \in \mathbb{R}^{N\times M}$ *for* $k = 1, 2, ..., K$ *such that*

$$\mathcal{K} = \sum_{k=1}^{k} TT_{(1,2,4,3)}(\mathcal{D}^{(k)} \odot_E \mathbf{P}^{(k)})$$
$$s.t. \quad \mathcal{K}_{i,j,m,n} = \sum_{k=1}^{k} \mathbf{P}^{(k)}_{n,m} \cdot \mathcal{D}^{(k)}_{i,j,n}.$$

□

Figure 5 illustrates the relation between standard convolution and rank-k FALCON. For each $k = 1, 2, ..., K$, we construct the tensor $\mathcal{K}^{(k)}$ using EHP of the depthwise convolution kernel $\mathcal{D}^{(k)}$ and the pointwise convolution kernel $\mathbf{P}^{(k)}$. Then, we construct the standard kernel $\mathcal{K}$ by the element-wise sum of the tensors $\mathcal{K}^{(k)}$ for all k.

3.5 Quantitative Analysis

In this section, we evaluate the compression and the computation reduction of FALCON and rank-k FALCON. All the analysis is based on one convolution layer. The comparison of the numbers of parameters and FLOPs of FALCON and other competitors is in Appendix B.

3.5.1 Falcon

We analyze the compression and the computation reduction rates of FALCON in Theorems 3 and 4.

Theorem 3 *Compression Rate (CR) of* FALCON *is given by*

$$CR = \frac{\#\text{ of parameters in standard convolution}}{\#\text{ of parameters in FALCON}} = \frac{D^2MN}{MN + D^2N}$$

where D^2 is the size of standard kernel, M is the number of input channels, and N is the number of output channels.

Proof 3.5.1 *Standard convolution kernel has D^2MN parameters.* FALCON *includes pointwise convolution and depthwise convolution which requires MN and D^2N parameters, respectively. Thus, the compression rate of* FALCON *is* $CR = \frac{D^2MN}{MN + D^2N}$. □

Theorem 4 *Computation Reduction Rate (CRR) of* FALCON *is described as:*

$$\begin{aligned} CRR &= \frac{\#\text{ of FLOPs in standard convolution}}{\#\text{ of FLOPs in FALCON}} \\ &= \frac{H'W'MD^2N}{HWMN + H'W'D^2N} \end{aligned}$$

where H' and W' are the height and the width of output, and H and W are the height and the width of input.

Proof 3.5.2 *The standard convolution operation requires $H'W'D^2MN$ FLOPs [11].* FALCON *includes pointwise convolution and depthwise convolution. Pointwise convolution has kernel size $D = 1$ with stride $s = 1$ and no padding, so the intermediate tensor $\mathfrak{O}'$ has the same height and width as the input feature maps. Thus, pointwise convolution needs $HWMN$ FLOPs. Depthwise convolution has the number of input channel $M = 1$, so it needs $H'W'D^2N$ FLOPs. The total FLOPs of* FALCON *is $HWMN + H'W'D^2N$, thus the computation reduction rate of* FALCON *is* $CRR = \frac{H'W'D^2MN}{HWMN + H'W'D^2N}$. □

3.5.2 Rank-k Falcon

We analyze the compression and computation reduction rates of rank-k FALCON in Theorem 5.

Theorem 5 *Compression Rate (CR_k) and Computation Reduction Rate (CRR_k) of rank-k* FALCON *are described as:*

$$CR_k = \frac{CR}{k} \qquad CRR_k = \frac{CRR}{k}$$

Proof 3.5.3 *The numbers of parameters and FLOPs increase for k times since rank-k* FALCON *duplicates* FALCON *for k times. Thus, the compression rate and the computation reduction rate are calculated as* $CR_k = \frac{CR}{k}$ *and* $CRR_k = \frac{CRR}{k}$. □

Chapter 4

Experiments

We validate the performance of FALCON through extensive experiments. We aim to answer the following questions:

- **Q1. Accuracy vs. Compression (Section 4.3).** What are the accuracy and the compression tradeoffs of FALCON and competitors? Which method gives the best accuracy for a given compression rate?
- **Q2. Accuracy vs. Computation (Section 4.4).** What are the accuracy and the computation tradeoffs of FALCON and competitors? Which method gives the best accuracy for a given amount of computation?
- **Q3. Rank-k Falcon (Section 4.5).** How do the accuracy, the number of parameters, and the number of FLOPs change as the rank k increases in FALCON?

4.1 Experimental Setup

Datasets. We perform image classification task on four famous datasets - CIFAR10, CIFAR100, SVHN, and ImageNet. CIFAR10 and CIFAR100 are subsets of the 80 million tiny images dataset [12]. SVHN [13] is a real-world dataset containing images of house address numbers obtained from Google Street View. ImageNet ILSVRC2012 [14] is a large color image database with hand-annotated label. Detailed information of these datasets is described in

Table 2.

Table 2: Datasets.

dataset	# of classes	input size	# of train	# of test
CIFAR-10[1]	10	$32 \times 32 \times 3$	10×6000	10000
CIFAR-100[2]	100	$32 \times 32 \times 3$	100×600	10000
SVHN[3]	10	32×32	73257	26032
ImageNet[4]	1000	$224 \times 224 \times 3$	1.2×10^6	150000

Models. For CIFAR10, CIFAR100, and SVHN datasets, we choose VGG19 and ResNet34 to evaluate the performance. We shrink the sizes of both models since the sizes of these three datasets are smaller compared to that of Imagenet. In VGG19, we reduce the number of fully connected layers and the number of features in fully connected layers: three large fully connected layers (4096-4096-1000) in VGG19 are replaced with two small fully connected layers (512-10 or 512-100). In ResNet34, we remove the first 7×7 convolution layer and max-pooling layer since the input size (32×32) of these datasets is smaller than the input size (224×224) of ImageNet. On both models, we replace all standard convolution layers (except for the first convolution layer) with those of FALCON or other competitors in order to compress and accelerate the model.

For ImageNet, we choose VGG16_BN (VGG16 with batch normalization after every convolution layer) and ResNet18. We use the pretrained model from Pytorch model zoo as the baseline model with standard convolution, and replace the standard convolution with other types of convolutions.

Implementation. We construct all models using Pytorch framework. All

[1] `https://www.cs.toronto.edu/~kriz/cifar.html`
[2] `https://www.cs.toronto.edu/~kriz/cifar.html`
[3] `http://ufldl.stanford.edu/housenumbers/`
[4] `http://www.image-net.org`

the models are trained and tested on GeForce GTX 1080 Ti GPU.

4.2 Fitting Convolution Unit into Model

We evaluate the performance of FALCON compared to DSConv, MobileConvV2, and ShuffleUnit. We take each standard convolution layer (StConv) as a unit, and replace StConv with those from FALCON or other competitors. We evaluate the classification accuracy, the number of parameters in the model, and the number of FLOPs needed for forwarding one image.

Falcon When replacing StConv with FALCON, we use the same setting as StConv. If there are BN and ReLU after StConv, we add BN and ReLU at the end of FALCON; if there is only ReLU after StConv, we only add ReLU at the end of FALCON. This is because FALCON is initialized by approximating StConv kernel by EHP. Using the same setting of BN and ReLU as StConv is more efficient for FALCON to approximate the StConv.

We initialize the pointwise convolution kernel and the depthwise convolution kernel of FALCON by approximating the pretrained standard convolution kernel using EHP. The approximation process is as follows: 1) we first initialize the pointwise convolution kernel and the depthwise convolution kernel randomly, and 2) the pointwise convolution kernel and the depthwise convolution kernel are updated using gradient descent such that the mean squared error of their EHP product and the standard convolution kernel is minimized. Rank-k FALCON uses the same initialization method.

Table 3: Performance of FALCON compared to StConv, DSConv, MobileConvV2, and ShuffleUnit on CIFAR10/CIFAR100 datasets. Bold font indicates the best accuracy among competing compression methods. FALCON gives the highest accuracy with the similar number of parameters and FLOPs compared to other methods.

(a) VGG19-CIFAR10

ConvType	Accuracy	# of param	# of FLOPs
StConv	93.28%	20.30M	398.70M
FALCON	**93.23**%	2.56M	47.23M
DSC	91.76%	2.56M	48.02M
MobileConvV2-0.5	92.58%	2.67M	51.80M
ShuffleUnit 2×(g=2)	91.77%	2.74M	46.66M

(b) ResNet34-CIFAR10

ConvType	Accuracy	# of param	# of FLOPs
StConv	93.27%	21.29M	292.52M
FALCON	**92.86**%	2.63M	46.33M
DSC	91.30%	2.62M	38.41M
MobileConvV2-0.5	90.61%	2.55M	39.78M
ShuffleUnit 2×(g=2)	90.28%	3.08M	49.78M

(c) VGG19-CIFAR100

ConvType	Accuracy	# of param	# of FLOPs
StConv	71.93%	20.35M	398.75M
FALCON	**71.55**%	2.61M	47.28M
DSC	68.47%	2.61M	48.07M
MobileConvV2-0.5	68.31%	2.71M	51.85M
ShuffleUnit 2×(g=2)	69.31%	2.79M	46.71M

(d) ResNet34-CIFAR100

ConvType	Accuracy	# of param	# of FLOPs
StConv	66.79%	21.34M	292.57M
FALCON	**68.99**%	2.67M	46.38M
DSC	65.47%	2.67M	38.45M
MobileConvV2-0.5	59.78%	2.59M	39.83M
ShuffleUnit 2×(g=2)	65.25%	3.17M	49.88M

Table 4: Performance of FALCON compared to StConv, DSConv, MobileConvV2, and ShuffleUnit on SVHN dataset. Bold font indicates the best accuracy among competing compression methods. FALCON gives the highest accuracy with the similar number of parameters and FLOPs compared to other methods.

(a) VGG19-SVHN

ConvType	Accuracy	# of param	# of FLOPs
StConv	94.92%	20.30M	398.70M
FALCON	**94.22**%	2.56M	47.23M
DSC	94.00%	2.56M	48.02M
MobileConvV2-0.5	93.11%	2.67M	51.80M
ShuffleUnit 2×(g=2)	93.54%	2.74M	46.66M

(b) ResNet34-SVHN

ConvType	Accuracy	# of param	# of FLOPs
StConv	94.07%	21.29M	292.52M
FALCON	**94.03**%	2.63M	46.33M
DSC	88.62%	2.62M	38.41M
MobileConvV2-0.5	90.38%	2.55M	39.78M
ShuffleUnit 2×(g=2)	92.99%	3.08M	49.78M

Table 5: Performance of FALCON compared to StConv, DSConv, MobileConvV2, and ShuffleUnit on ImageNet dataset. Bold font indicates the best accuracy among competing compression methods. FALCON provides the highest accuracy with similar number of parameters and FLOPs compared to other methods.

(a) VGG16_BN

ConvType	Top-1 Accuracy	Top-5 Accuracy	# of param	# of FLOPs
StConv	73.37%	91.50%	138.37M	15484.82M
FALCON	**71.63**%	**90.57**%	125.33M	1950.75M
DSC	70.34%	89.71%	125.33M	1989.49M
MobileConvV2-0.5	67.80%	87.90%	125.44M	2180.49M
ShuffleUnit 2×(g=2)	70.40%	89.84%	125.77M	2014.73M

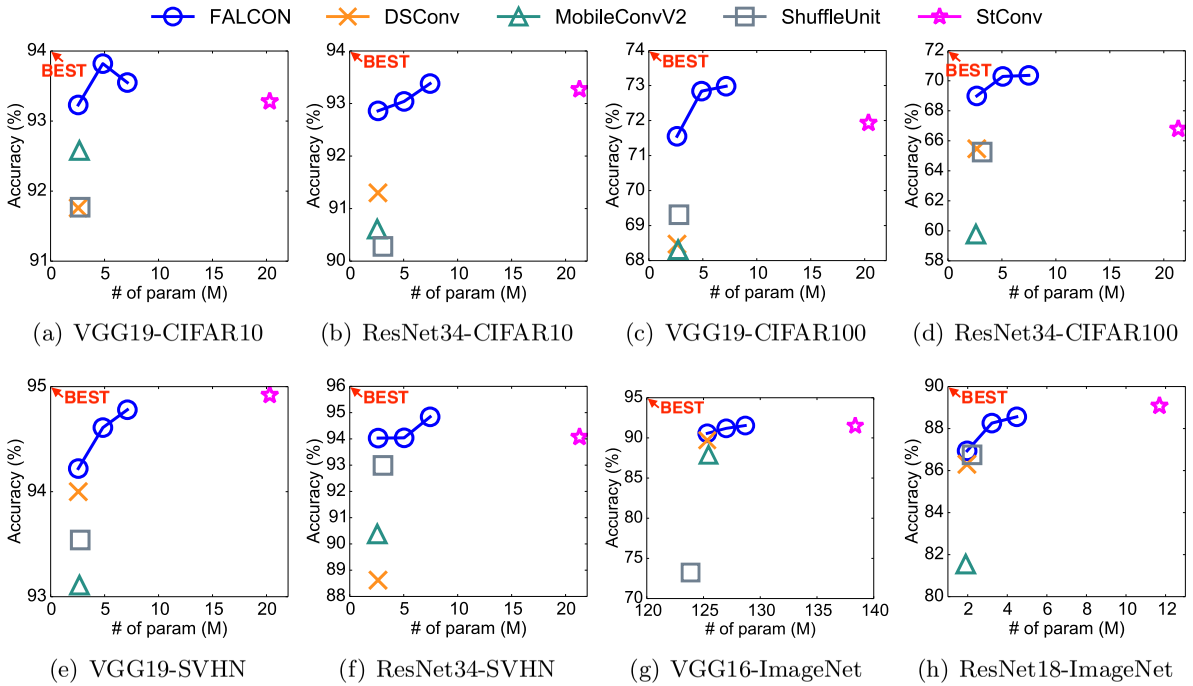
(b) ResNet18

ConvType	Top-1 Accuracy	Top-5 Accuracy	# of param	# of FLOPs
StConv	69.76%	89.08%	11.69M	1814.07M
FALCON	**66.13**%	**86.94%**	1.97M	395.40M
DSC	65.30%	86.30%	1.96M	336.81M
MobileConvV2-0.5	58.99%	81.55%	1.90M	340.06M
ShuffleUnit 2×(g=2)	65.73%	86.75%	2.22M	438.89M

DSConv DSConv (shown in Figure 2(c)) has the most similar architecture as FALCON among competitors, and thus DSConv has nearly the same number of parameters as that of FALCON. As in the setting of FALCON, the existence of BN and ReLU at the end of DSConv depends on that of StConv.

MobileConvV2 In MobileConvV2 architecture (shown in Figure 2(d)), we adjust the numbers of parameters and FLOPs by changing the expansion ratio t as described in Section 2.3, which is represented as 'MobileConvV2-t'. We choose $t = 0.5$ as the baseline MobileConvV2 to compare with FALCON, since two pointwise convolutions bring lots of parameters and FLOPs to MobileConvV2.

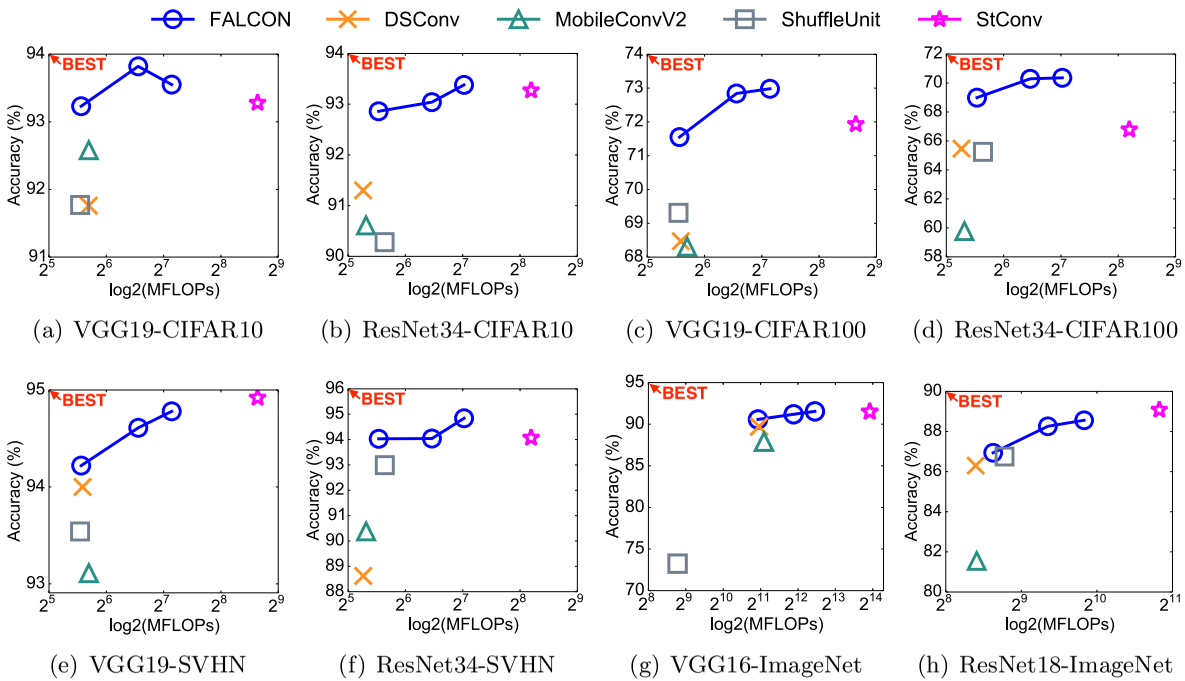
ShuffleUnit In ShuffleUnit (shown in Figure 2(e)), we adjust the numbers of parameters and FLOPs by changing the width multiplier α [7] and the number of groups g, which is represented as 'ShuffleUnit $\alpha\times$(g=g)'. Note that the width multiplier is used to adjust the number of input channels M and the number of output channels N of a convolution layer; if the width multiplier is α, the numbers of input and output channels become αM and αN. While experimenting with ResNet, we find that ShuffleUnit does not cooperate well with ResNet: ResNet34 with ShuffleUnit does not converge. We suspect that residual block and ShuffleUnit may conflict with each other because of redundant residual connections: the gradient may not find the right path towards previous layers. For this reason, we delete the shortcut of all residual blocks in ResNet34 when using ShuffleUnit.



(a) VGG19-CIFAR10 (b) ResNet34-CIFAR10 (c) VGG19-CIFAR100 (d) ResNet34-CIFAR100

(e) VGG19-SVHN (f) ResNet34-SVHN (g) VGG16-ImageNet (h) ResNet18-ImageNet

Figure 6: Accuracy w.r.t. number of parameters on different models and datasets. The three blue circles correspond to rank-1, 2, 3 FALCON (from left to right order), respectively. FALCON provides the best accuracy for a given number of parameters.



(a) VGG19-CIFAR10 (b) ResNet34-CIFAR10 (c) VGG19-CIFAR100 (d) ResNet34-CIFAR100

(e) VGG19-SVHN (f) ResNet34-SVHN (g) VGG16-ImageNet (h) ResNet18-ImageNet

Figure 7: Accuracy w.r.t. FLOPs on different models and datasets. The three blue circles correspond to rank-1, 2, 3 FALCON (from left to right order), respectively. FALCON provides the best accuracy for a given number of FLOPs.

4.3 Accuracy vs. Compression

We evaluate the accuracy and the compression rate of FALCON and its competitors. Tables 3, 4 and 5 show the accuracies and the compression rates of methods on four image datasets. Note that FALCON provides the highest accuracy for all datasets while using similar or smaller number of parameters than competitors. Specifically, FALCON achieves up to 8× compression rates with a tiny drop in accuracy by less than 1%, compared to the standard convolution (StConv). Figure 6 shows the tradeoff of accuracy and the number of parameters. Note that FALCON shows the best tradeoff between accuracy and compression, giving the highest accuracy with similar compression rates.

4.4 Accuracy vs. Computation

We evaluate the accuracy and the amount of computation of FALCON and its competitors. We use the number of multiply-adds floating point operations (FLOPs) needed for forwarding one image to a model as the metric of computation. Tables 3, 4 and 5 also show the accuracies and the number of FLOPs of methods on four image datasets. Note that FALCON provides the highest accuracy for all datasets while using similar FLOPs than competitors in most cases. Compared to the standard convolution (StConv), FALCON achieves up to 8× FLOPs reduction across different models on different datasets.

We also notice from Table 5(a) that ShuffleUnit needs less number of FLOPs while using similar number of parameters as other models. This is because ShuffleUnit changes pointwise convolution into group 1×1 convolution, which reduces the numbers of parameters and FLOPs in convolution layers by

the number of group times, which is 4×. In VGG16_BN, the number of parameters is dominated by those in fully-connected layers, and the number of FLOPs is dominated by computations in convolution layers, which makes the reduction of FLOPs in Table 5(a). Figure 7 shows the tradeoff of accuracy and the number of FLOPs. Note that FALCON shows the best tradeoff between accuracy and computation, giving the highest accuracy with similar number of FLOPs.

4.5 Rank-k Falcon

We evaluate the performance of rank-k FALCON by increasing the rank k and monitoring the changes in the numbers of parameters and FLOPs. In Table 6, 7 and 8, we observe three trends as the rank k increases: 1) the accuracy continues to increase, 2) the number of parameters increases, and 3) the number of floating point operations (FLOPs) increases. Especially, we note that when the rank k is 3, FALCON often gives even higher accuracy than the standard convolution, while using smaller number of parameters and FLOPs. For example, rank-3 FALCON applied to ResNet34 on CIFAR100 dataset shows 3.57 percentage point higher accuracy, with 2.8× smaller number of parameters and 2.25× smaller number of FLOPs compared to the standard convolution. Thus, rank-k FALCON is a versatile method to further improve the accuracy of FALCON while sacrificing a bit of compression and computation.

Table 6: Performance of rank-k FALCON on CIFAR10/CIFAR100 datasets.

(a) VGG19-CIFAR10

ConvType	Accuracy	# of param		# of FLOPs	
StConv	93.28%	20.30M		398.70M	
FALCON-k1	93.23%	2.56M	(7.93×)	47.23M	(8.44×)
FALCON-k2	93.82%	4.84M	(4.19×)	94.20M	(4.23×)
FALCON-k3	93.50%	7.11M	(2.86×)	141.16M	(2.82×)

(b) ResNet34-CIFAR10

ConvType	Accuracy	# of param		# of FLOPs	
StConv	93.27%	21.29M		292.52M	
FALCON-k1	92.86%	2.63M	(8.10×)	46.33M	(6.31×)
FALCON-k2	93.04%	5.04M	(4.22×)	88.21M	(3.32×)
FALCON-k3	93.38%	7.45M	(2.86×)	130.08M	(2.25×)

(c) VGG19-CIFAR100

ConvType	Accuracy	# of param		# of FLOPs	
StConv	71.93%	20.35M		398.75M	
FALCON-k1	71.55%	2.61M	(7.80×)	47.28M	(8.43×)
FALCON-k2	72.84%	4.88M	(4.17×)	94.24M	(4.23×)
FALCON-k3	72.98%	7.16M	(2.84×)	141.21M	(2.82×)

(d) ResNet34-CIFAR100

ConvType	Accuracy	# of param		# of FLOPs	
StConv	66.79%	21.34M		292.57M	
FALCON-k1	68.99%	2.67M	(7.99×)	46.38M	(6.31×)
FALCON-k2	70.29%	5.08M	(4.20×)	88.25M	(3.32×)
FALCON-k3	70.36%	7.49M	(2.85×)	130.13M	(2.25×)

Table 7: Performance of rank-k FALCON on SVHN dataset.

(a) VGG19-SVHN

ConvType	Accuracy	# of param		# of FLOPs	
StConv	94.92%	20.30M		398.70M	
FALCON-k1	94.22%	2.56M	(7.93×)	47.23M	(8.44×)
FALCON-k2	94.61%	4.84M	(4.19×)	94.20M	(4.23×)
FALCON-k3	94.78%	7.11M	(2.86×)	141.16M	(2.82×)

(b) ResNet34-SVHN

ConvType	Accuracy	# of param		# of FLOPs	
StConv	94.07%	21.29M		292.52M	
FALCON-k1	94.03%	2.63M	(8.10×)	46.33M	(6.31×)
FALCON-k2	94.04%	5.04M	(4.22×)	88.21M	(3.32×)
FALCON-k3	94.84%	7.45M	(2.86×)	130.08M	(2.25×)

Table 8: Performance of rank-k FALCON on ImageNet dataset.

(a) VGG16_BN

ConvType	Top-1 Accuracy	Top-5 Accuracy	# of param		# of FLOPs	
StConv	73.37%	91.50%	138.37M		15484.82M	
FALCON-k1	71.63%	90.57%	125.33M	(1.10×)	1950.75M	(7.94×)
FALCON-k2	72.88%	91.19%	127.00M	(1.09×)	3777.86M	(4.10×)
FALCON-k3	73.24%	91.54%	128.68M	(1.08×)	5604.97M	(2.76×)

(b) ResNet18

ConvType	Top-1 Accuracy	Top-5 Accuracy	# of param		# of FLOPs	
StConv	69.76%	89.08%	11.69M		1814.07M	
FALCON-k1	66.13%	86.94%	1.97M	(5.93×)	395.40M	(4.59×)
FALCON-k2	68.03%	88.26%	3.22M	(3.63×)	653.00M	(2.78×)
FALCON-k3	69.07%	88.56%	4.48M	(2.61×)	910.61M	(1.99×)

Chapter 5

Related Works

Over the past several years, a lot of studies focused on compressing and accelerating DNN to reduce model size, runtime, and energy consumption.

It is believed that DNNs are over-parameterized. Pruning [15, 16] aims at removing useless weights or setting them to zero. Weight-sharing [17, 18, 19, 20, 21] is a common compression method. It is applied to DNNs' weights by storing only assignments and centroids of weights. While using the model, weights are loaded according to assignments and centroids. Although pruning and weight-sharing can significantly reduce the model size, they are not efficient in reducing the amount of computation. Quantizing [22, 23, 24, 25] the model into binary or ternary weights reduces model size and computation simultaneously: replacing arithmetic operations with bit-wise operations remarkably accelerates the model.

Layer-wise approaches are also employed to efficiently compress the model. A typical example of such approaches is low-rank approximation [26, 27, 28]; it treats the weights as a tensor and uses general tensor approximation methods to compress the tensor. To reduce computation, approximation methods should be carefully chosen, since some of approximation methods may increase computation of the model.

Compressing existing models has limitations since they are originally designed to be deep and large to give high accuracy. A recent trend is to de-

sign a brand new architecture that is small and efficient. Mobilenet [7], MobilenetV2 [8], and Shufflenet [9] are the most representative approaches, and they use depthwise convolution and pointwise convolution as building blocks for designing convolution layers. Our proposed FALCON gives a thorough interpretation of depthwise convolution and pointwise convolution, and applies them into model compression, giving the best accuracies with regard to compression and computation.

Chapter 6

Conclusion

We propose FALCON, an accurate and lightweight convolution method to replace standard convolution.

By interpreting existing convolution methods based on depthwise separable convolution using EHP operation, FALCON and its general version rank-k FALCON provide accurate and efficient compression methods on CNN.

Extensive experiments show that FALCON gives the best accuracy for a given number of parameter or computation, outperforming other convolution models based on depthwise separable convolution. Compared to the standard convolution, FALCON gives up to $8\times$ compression and $8\times$ computation reduction while giving similar accuracy. We also show that rank-k FALCON provides better accuracy than the standard convolution, while using smaller numbers of parameters and computations.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[3] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI, 2017*, pp. 4278–4284, 2017.

[4] W. Yin, H. Schütze, B. Xiang, and B. Zhou, "ABCNN: attention-based convolutional neural network for modeling sentence pairs," *TACL*, vol. 4, pp. 259–272, 2016.

[5] D. H. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Recsys, 2016*, pp. 233–240, 2016.

[6] L. Sifre, *Rigid-motion scattering for image classification.* PhD thesis, École Polytechnique, 2014.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR, abs/1704.04861, 2017. 2, 4, 5, 6*, 2017.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[9] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," 2017.

[10] Chollet and François, "Xception: Deep learning with depthwise separable convolutions," 2016. cite arxiv:1610.02357.

[11] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *Neural Information Processing Systems (NIPS)*, 2017.

[12] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.

[13] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[15] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *Neural Information Processing Systems (NIPS)*, 2014.

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *International Conference on Learning Representations (ICLR)*, 2016.

[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *International Conference on Learning Representations (ICLR)*, 2016.

[18] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," *International Conference on Learning Representations (ICLR)*, 2017.

[19] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," *International Conference on Machine Learning (ICML)*, 2015.

[20] Y. Choi, M. El-Khamy, and J. Lee, "Towards the limit of network quantization," *International Conference on Learning Representations (ICLR)*, 2017.

[21] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," *Neural Information Processing Systems (NIPS)*, 2017.

[22] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Neural Information Processing Systems (NIPS)*, 2015.

[23] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training neural networks with weights and activations constrained to +1 or 1," *Neural Information Processing Systems (NIPS)*, 2016.

[24] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," *International Conference on Learning Representations (ICLR)*, 2017.

[25] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *International Conference on Learning Representations (ICLR)*, 2017.

[26] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *International Conference on Learning Representations (ICLR)*, 2015.

[27] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *International Conference on Learning Representations (ICLR)*, 2016.

[28] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," *Neural Information Processing Systems (NIPS)*, 2015.

A Generality of EHP

We show EHP is a key operation to understand additional convolution architectures MobilenetV2 and Shufflenet which are based on depthwise separable convolution.

MobilenetV2. As shown in Figure 2(d), MobilenetV2 has an additional pointwise convolution before depthwise convolution in Mobilenet: one layer of MobilenetV2 consists of two pointwise convolutions and one depthwise convolution. In another point of view, MobilenetV2 can be understood as FALCON followed by additional pointwise convolution. I.e., MobilenetV2 performs EHP operation as FALCON does, and performs additional pointwise convolution after that.

Shufflenet. As shown in Figure 2(e), Shufflenet consists of depthwise convolution and pointwise group convolution which is a variant of pointwise convolution. We represent the convolution layer of Shufflenet using EHP as follows. Let g be the number of groups. We divide the standard convolution kernel $\mathcal{K} \in \mathbb{R}^{D\times D\times M\times N}$ into g group standard convolution kernels. Then, the relation of g-th group standard convolution kernel $\mathcal{K}^g \in \mathbb{R}^{D\times D\times \frac{M}{g}\times \frac{N}{g}}$ with regard to g-th depthwise convolution kernel $\mathcal{D}^g \in \mathbb{R}^{D\times D\times \frac{M}{g}}$ and g-th pointwise group convolution kernel $\mathbf{P}^g \in \mathbb{R}^{\frac{M}{g}\times \frac{N}{g}}$ is

$$\mathcal{K}^g = \mathcal{D}^g \odot_E \mathbf{P}^g \quad \text{s.t.} \quad \mathcal{K}^g_{i,j,m_g,n_g} = \mathcal{D}^g_{i,j,m_g} \cdot \mathbf{P}^g_{m_g,n_g}$$

where $m_g = 1, 2, ..., \frac{M}{g}$ and $n_g = 1, 2, ..., \frac{N}{g}$. Each group standard convolution is equivalent to the combination of a depthwise convolution and a pointwise convolution, and thus easily expressed with EHP as in Mobilenet.

Therefore, each layer of Shufflenet is equivalent to the layer consisting of one group convolution followed by standard convolution.

B Parameters and FLOPs

We summarize the numbers of parameters and FLOPs for FALCON and competitors in Table 9.

Table 9: the numbers of parameters and FLOPs of FALCON and competitors. Symbols are described in Table 1.

Convolution	**# of parameters**	**# of FLOPs**
FALCON	$MN + D^2N$	$HWMN + H'W'D^2N$
DSConv	$MN + D^2M$	$HWD^2M + H'W'MN$
MobilenetV2	$tM^2 + tD^2M + tMN$	$tHWM^2 + tH'W'D^2M + tH'W'MN$
Shufflenet	$\frac{1}{4}(\frac{MN}{g} + D^2N + \frac{N^2}{g})$	$\frac{1}{4}(\frac{HWMN}{g} + H'W'D^2N + \frac{H'W'N^2}{g})$
Standard convolution	$MN + D^2N$	$H'W'D^2MN$

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. U Kang. Under his guidance, I have learnt not only the professional knowledge, but also how to be a good researcher. His support and assistance of my study and research are integral parts during my Master's degree period.

I would also like to acknowledge the rest of my thesis committee: Prof. Byung-Gon Chun and Prof. Gunhee Kim, for dedicating time to review my paper. Their insightful comments are really helpful for me to improve my research.

My sincere thanks also goes to Mr. Jun-Gi Jang, for his collaboration and assistance on this research. As a co-worker and a senior, he has taught me how to do research and write papers in detail, which is very helpful.

My heartfelt thanks to my fellow labmates, for helping me get through all difficulties in foreign country, for creating a great study atmosphere in the lab, and for always being there and sharing happiness and difficulties. The lab makes me feel right at home.

Last but not the least, I would like to acknowledge my family. Without their support and encouragement during my Master's degree, I would not have today's accomplishment.